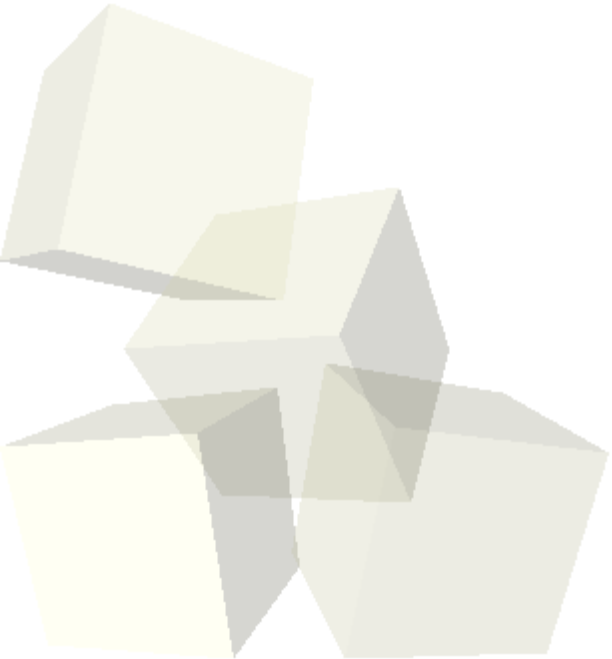
Scope and Problem Decomposition

9-12-2006

Opening Discussion

- What did we talk about last class?
- Why do we want to use functions in our code?
- Do you have any questions about the reading?



Scope

- Everything you declare in a programming language has a range of statements that it can be accessed over. This is called its scope.
- Scoping in C is extremely simple. There are two types of scope, local and global.
  - Global scope means something is visible anywhere between the declaration and the end of the file. Anything that isn't declared inside a code block (inside curly braces) is global. This includes all functions and any variables you declare outside the functions (generally a bad idea).
  - Local scope goes from the point of declaration down to the end of the code block it was declared in. Function parameters have local scope.

Why to Scope?

- Small scopes reduce complexity. Without them, programs quickly become a tangled mess. They also allow you to reuse names. Without local scoping every variable you ever create would have to have a unique name.
- There are times, as we saw last class, when global scope seems to make things easier. However, this is an illusion created by small programs. The bigger the program gets, the more problems you will run into with global variables.

Scoping and the Call Stack

- It is worth taking a minute to discuss a detail of local variables in C, they way they are implemented in memory. This will be very significant a bit later.
- Every time you call a function, that function gets a chunk of memory to hold the local variables. When the function returns, that chunk is freed.
- The chunks are laid end-to-end so they resemble things being stacked up. Like a stack of books, you can only easily access the ones in the top chunk.
- Let's draw this out on the board.

Problem Decomposition

- Let's solve a little problem now. We'll use one of my favorite problems, making a ray tracer. We will only do a little part of it though, but I want to show how we can take a large problem and break it into pieces.
- When breaking problems into pieces, one of the concerns that should be high on your mind is whether the pieces are reusable. An ideal subdivision will make functions that you can use over and over again.

Conditional Execution

- As you have probably noticed, it's been challenging to find problems that you can do with what we have talked about. This is because every line of code you write always gets executed in exactly the same order. All that can change is the values the user inputs.
- Conditional execution is the concept that you have have part of the code where certain lines happen under one condition and other lines happen in a different condition. This opens the door for much more complex problems and a lot more flexibility in what we solve.

The if Statement

- The most common form of conditional execution is the if statement. It basically says, do something if a certain condition is true. There can also be an else part for what to do when the condition is false.
- What is done in if or else can be any C statement. Normally it is a compound statement, many statements inside of curly braces. Single statements can be used though some programmers strictly avoid this.
- Let's think of how we might use an if statement and add one to our ray tracing code.

Minute Essay

- What are the primary problems caused by using global scope for variable? Can you think of any issues with global scope that I haven't mentioned yet?
- Quiz #2 is on Thursday, and assignment #2 is due a week from today.